Amended Claims:

Claims 1-11 (cancelled)

12. (Currently amended) A method of compacting an intermediate ~~programme~~program consisting of a sequence of standard instructions, used in an on-board system, said on-board system being provided with a memory and a ~~programme~~program language interpreter capable of turning the intermediate ~~programme~~program into instructions of an object code that can be run directly by a microprocessor, said method consisting in:

a) searching through said intermediate ~~programme~~program for identical sequences of successive standard instructions;

b) subjecting said identical sequences of successive instructions to a comparison test to find a function, based on at least the number of occurrences of these sequences in said intermediate ~~programme~~program, that is higher than a reference value and, if the test returns a positive response, for each identical sequence of successive standard instructions which satisfies said test step,

B1

c) generating a specific instruction by defining a specific operating code and associating said specific operating code with the sequence of successive standard instructions which satisfied said test,

d) replacing each occurrence of each sequence of standard successive instructions in said intermediate ~~programme~~program with said specific operating code associated with it to obtain a compacted intermediate ~~programme~~program, consisting of a series of standard instructions and specific operating codes, and

e) storing in said memory an execution table which enables a reciprocal link to be established between each specific operating code inserted and the sequence of successive standard instructions associated with the latter, said program language interpreter being adapted to determine whether a read code value corresponds to a standard type code or to a specific type code, making it possible to execute specific instructions, by calling on said execution table,

thereby enabling the memory space occupied by said compacted intermediate ~~programme~~program to be ~~optimised~~optimised by storing only one occurrence of said identical sequences of successive standard instructions in said memory.

13. (Previously presented) A method as claimed in claim 12, wherein said function is also a function of the size of each identical sequence of successive instructions.

14. (Currently amended) A method as claimed in claim 12, wherein in order to compress a plurality of intermediate ~~programme~~programs, said method also consists in:

- storing said execution table relating to at least one compacted intermediate ~~programme~~program and, for every additional intermediate ~~programme~~program subjected to a compaction process,

- reading said stored execution table, and

- running the compaction process for every additional ~~programme~~program, taking account of the specific codes and instructions stored in said execution table.

15. (Currently amended) A method of running a compacted intermediate ~~programme~~program obtained by applying a compaction method ~~as claimed in claim 1~~, said compacted intermediate ~~programme~~program consisting of a succession of standard instructions and specific operating codes stored in the memory of an on-board system, wherein said method consists in:

- ~~recognis~~recognizing in said memory the existence of a stored execution table containing at least one sequence of successive instructions associated with a specific operating code by means of a reciprocal link;

- calling up a command, via a ~~said programme~~program language interpreter, to read the successive standard instructions or specific operating codes of said compacted intermediate ~~programme~~program and, in the presence of a specific operating code:

- retrieving said sequence of successive instructions associated with said specific operating code from the memory by means of a read instruction and, in the presence of a standard instruction,

- commanding the execution of said standard instruction by means of a read instruction,

said program language interpreter being adapted to determine whether a read code value corresponds to a standard type code or to a specific type code, making it possible to execute specific instructions, by calling on said execution table.

16. (Currently amended) A method as claimed in claim 15, wherein if a sequence of successive instructions associated with a specific operating code is called up, the current value of a ~~programme~~program counter is incremented in a stack associated with the specific operating codes and a ~~programme~~program pointer points to the first instruction of said sequence of specific instructions, after which, on running an instruction to end the sequence of specific instructions, said ~~programme~~program counter is decremented and the execution process continues starting with the next instruction or specific operating code.

17. (Previously presented) A method as claimed in claim 16, wherein the stack associated with the specific operating codes and the stack associated with the standard instructions are a single stack.

18. (Currently amended) A multi-application on-board system comprising computing resources, a memory and language interpreter capable of turning an intermediate ~~programme~~program into instructions which are directly executable by the computing resources, wherein said multi-application on-board system also at least comprises:

- one table of standard codes constituting said intermediate ~~programme~~program stored at the level of said language interpreter;

- at least one compacted intermediate ~~programme~~program constituting an application and consisting of a series of specific instruction codes and standard instruction codes, said specific instruction codes corresponding to sequences of successive standard instructions;

- an execution table enabling a reciprocal link to be established between an operating specific instruction code and the sequence of successive standard instructions associated with the latter, said at least one compacted intermediate ~~programme~~program and said execution table being stored in said memory, thereby enabling the memory space occupied by said compacted intermediate ~~programme~~program to be ~~optimised~~optimised by storing in said programmable memory only one occurrence of said identical sequences of successive instructions, said program language interpreter being adapted to determine whether a read code value corresponds to a standard type code or to a specific type code, making it possible to execute specific instructions, by calling on said execution table.~~.~~

19. (Previously presented) An on-board system as claimed in claim 18, wherein said execution table comprises at least:

- a file of successive sequences corresponding to said specific instruction codes;

- a table of specific instruction codes and addresses at which said specific instruction codes are embedded in the table of successive sequences.

20. (Previously presented) An on-board system as claimed in claim 19, wherein said file of successive sequences corresponding to said specific instruction codes and said table of specific instruction codes are stored in a programmable memory of said on-board system.

21. (Currently amended) A compaction system for an intermediate ~~programme~~program, said intermediate ~~programme~~program consisting of a series of standard instructions which can be executed by a target unit, wherein said system comprises at least:

- means for ~~analys~~analyzing all the standard executable instructions enabling, by means of a reading process, said intermediate ~~programme~~program to distinguish between and establish

a list of all the sequences of executable standard instructions contained in said intermediate ~~programme~~program;

- means for counting the number of occurrences in this intermediate ~~programme~~program of each of the sequences of executable standard instructions forming part of said list;

- means for allocating to at least one sequence of executable standard instructions a specific code associated with this sequence of executable standard instructions in order to generate a specific instruction;

- means for replacing, in said intermediate ~~programme~~program, each occurrence of said sequence of executable standard instructions with said specific code associated with this sequence of executable standard instructions, representative of said specific instruction,

- means for storing in said memory an execution table which enables a reciprocal link to be established between each specific operating code inserted and the sequence of successive standard instructions associated with the latter, enabling a program language interpreter to determine whether a read code value corresponds to a standard type code or to a specific type code, and making it possible to execute specific instructions, by calling on said execution table, thereby enabling a compacted ~~programme~~program to be generated comprising a succession of executable standard instructions and specific instructions.

22. (Previously presented) A compaction system as claimed in claim 21, wherein said means for allocating to at least one sequence of executable standard instructions a specific code associated with said sequence of executable standard instruction in order to generate a specific instructions comprises at least:

- means for computing the value of a function based on at least the length of and number of occurrences of said sequence of executable standard instructions, said function being representative of the compression gain for said sequence of executable standard instructions;

- means for comparing the value of said function with a threshold value and, if said comparison returns a positive response;

- means for writing to a file, with a reciprocal link, a specific code and this sequence of executable standard instructions in order to constitute said specific instruction.